# Algorithmic aspects of algebraic methods for graph isomorphism testing

M. Kaukič, Dept. of Mathematical Methods, FRI ŽU,
Veľký Diel, 01026 Žilina, Slovakia
e-mail: *mike@frcatel.fri.utc.sk*

February 2, 2008

**Abstract**

We present the implementation of an algorithm for graph isomorphism testing, based on ideas about number of walks (of sufficiently large length) between vertices, expanded for strongly regular graphs (SRG-s) by testing the local complements and values of determinants of their adjacency matrices. All known non-isomorphic SRG-s (with no more than 64 vertices) are distinguishable by this method.

**Keywords**: *graph isomorphism, adjacency matrix, strongly regular graphs, local complement, Python, SciPy*

## 1 Introduction, basic algorithm

The graph isomorphism problem is one of difficult central problems in graph theory. It can be easily formulated as the question, whether two differently looking images represent the same graph. We will consider only undirected graphs without loops and multiple edges. Let us denote by $V(G), E(G)$ the sets of vertices and edges of graph $G$. Two graphs $G, H$ are isomorphic iff there exists a bijective mapping $f : V(G) \mapsto V(H)$ between sets of vertices of graphs $G$ and $H$ such that for all pairs of vertices $u, v \in V(G)$ the edge $\{u, v\}$ belongs to $E(G)$ if and only if $\{f(u), f(v)\}$ is an edge of graph $H$.

The *graph invariant* $\lambda$ is a mapping from the class of all graphs $\mathcal{G}$ to the set $\mathcal{R}$ such that for any isomorphic graphs $G, H$ the images $\lambda(G), \lambda(H)$ are the same, i.e. $\lambda(G) = \lambda(H)$. Commonly used graph invariants are e.g. number of vertices, number of edges, degree-sequence (the non-descending sequence of degrees of the vertices), eigenvalues or characteristic polynomial of the adjacency matrix, etc. All above-mentioned invariants are *incomplete*, which means that we can find pairs of nonisomorphic graphs with the same invariants.

There are also complete graph invariants, i.e. $\lambda(G) = \lambda(H)$ holds if and only if graphs $G, H$ are isomorphic (see e.g. [1] for greatest characteristic string invariant). Such invariants are not known to be computable in polynomial time, because the existence of polynomial algorithm for the class of graph isomorphism problems is still an open question. The existing polynomial algorithms are inexact in the sense that they cannot recognize some pairs of nonisomorphic graphs (because they use incomplete invariants for testing). Our proposed algorithm also belongs to this type of non-exact polynomial algorithms.

Let us briefly summarize the main results from Czimmermann [2], [3] where the basic variant of isomorphism testing algorithm using walks between vertices counting was formulated.

Let $A = (a_{ij})$ be the adjacency matrix of graph $G$. Then the number of all possible walks of length $k$ from vertex $v_i \in V(G)$ to vertex $v_j \in V(G)$ is simply the element $a_{ij}^{(k)}$ of matrix power $A^k$. Let us assign to every vertex $v_i \in V(G)$ the set

$$S_i = \{s_{i1}, s_{i2}, \ldots, s_{in}\}, \quad s_{ij} = \left\{a_{ij}^{(k)}\right\}_{k=1}^{m}, \tag{1}$$

i.e. $s_{ij}$ is the sequence formed from number of walks of length $1, 2, \ldots, m$ which begin in vertex $v_i$ and end in $v_j$. Let us have a pair of graphs $G, H$ with equal number $n$ of vertices for which the sequences of sets (1) are

$$\mathcal{W}_G = \{S_1, S_2, \ldots S_n\}, \ \mathcal{W}_H = \{T_1, T_2, \ldots, T_n\}.$$

If $G, H$ are isomorphic, then for suitable permutation of vertices of graph $G$ the two sequences are equal. The proposed algorithm is based on comparison of permuted sequences $\mathcal{W}_G, \mathcal{W}_H$ with $m$ (maximal length of walks considered) sufficiently large. In [3] it was shown that $m$ can be chosen as the number of common distinct eigenvalues of graphs $G, H$ (there is no additional information coming from walks of greater length). For this basic form of algorithm it is easy to find the causes when it fails; e.g. two nonisomorpic *strongly regular graphs* with the same set of parameters are indistinguishable. The graph $G$ is called *strongly regular (SRG) with parameters* $n, d, \alpha, \beta$ if it has $n$ vertices, every vertex has degree $d$, every pair of adjacent vertices has $\alpha$ common neighbours and every pair of distinct nonadjacent vertices has $\beta$ common neighbours. For example, the well-known Petersen graph (see Fig. 1) is SRG with parameters (10,3,0,1).

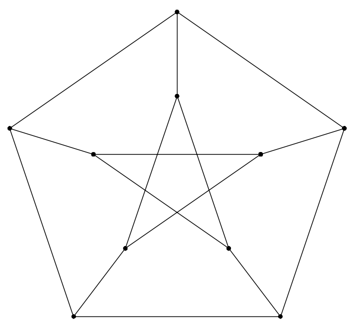

Figure 1: Petersen graph – strongly regular graph with parameters (10,3,0,1)

Each SRG has only three distinct eigenvalues, which are completely determined by parameters $n, d, \alpha, \beta$. Thus, the complete information about walks is contained in sequences of sets of walks with length $k \leq 3$. Such short walks are not sufficient to discover non-isomorfism of SRG-s with the same set of parameters. For SRG-s, we need to investigate some additional properties.

## 2 Improved algorithm for strongly regular graphs

The key idea is to destroy the regularity of given SRG by representing it with the sequence of suitable modified non-regular graphs. We will explore the notion of *local complement* (see [5]) for this purpose.

**Definition 2.1** *Let $G$ be a graph. The local complement $G_L(u)$ of $G$ at the vertex $u \in V(G)$ is defined to be the graph with the same vertex set as $G$ such that:*

1. *if $v$ and $w$ are distinct neighbours of $u$, then they are adjacent in $G_L(u)$ if and only if they are not adjacent in $G$,*
2. *if $v$ and $w$ are distinct vertices of $G$, and not both neighbours of $u$, then they are adjacent in $G_L(u)$ if and only if they are adjacent in $G$.*

Now, instead of testing isomorphism of two (SRG) graphs $G, H$, we can test for existence of suitable permutation of vertices of graph $H$ such that the sequences of local complements of $G, H$ have the same graph invariants (e.g. the sequences of sets of walks, mentioned in previous section).

| n | d | $\alpha$ | $\beta$ | No. |
|---|---|---|---|---|
| 16 | 6 | 2 | 2 | 2 |
| 25 | 12 | 5 | 6 | 15 |
| 26 | 10 | 3 | 4 | 10 |
| 28 | 12 | 6 | 4 | 4 |
| 29 | 14 | 6 | 7 | 41 |
| 35 | 16 | 6 | 8 | 3854 |
| 35 | 18 | 9 | 9 | 227 |

| n | d | $\alpha$ | $\beta$ | No. |
|---|---|---|---|---|
| 36 | 14 | 4 | 6 | 180 |
| 36 | 15 | 6 | 6 | 32548 |
| 37 | 18 | 8 | 9 | 6760 |
| 40 | 12 | 2 | 4 | 28 |
| 45 | 12 | 3 | 3 | 78 |
| 64 | 18 | 2 | 6 | 167 |

Table 1: Groups of nonisomorfic SRG-s

In Table 1 there are parameters and numbers of nonisomorphic SRG-s with given parameter sets for up to 64 vertices. We obtained adjacency matrices for all mentioned graphs from WEB-sources [7], [8]. To make the later processing easy, the SQL database of SRG-s were created with the aid of simple but powerful database engine *SQLite* (see [9]) and the corresponding interface module *pysqlite* [10] for programming language Python.

In the paper [6] the authors were able to distinguish all graphs from this table (although they have omitted two large groups of SRG-s with parameters $(35, 16, 6, 8)$ and $(37, 18, 8, 9)$) with their algorithm based on eigenvalues of certain matrix inspired by the notion of coined quantum walks. We will show that our modified algorithm can distinguish all of the graphs in Table 1, when combined with such easy to compute incomplete graph invariants as determinants of adjacency matrices of local complements.

## 3 Algorithm implementation and testing

We will focus on testing our algorithms only for set of SRG-s from Table 1, because strongly regular graphs cause main difficulties for graph isomorphism algorithms.

For the rapid implementation of basic algorithm (see section 1) we choose the interpreted programming language *Python* [12]. It has very clear syntax and together with *SciPy* (Scientific tools for Python,[13]) and IPython (enhanced Interactive Python shell, see [14]) it forms very powerful and convenient environment for rapid development of prototype applications.

In the basic algorithm we need to compute the powers $A^k$ of 0-1 matrices $A$ (for SRG-s, up to $k = 3$). Underlying C and FORTRAN libraries in SciPy use machine integer and floating point numbers with fixed precision. This will give wrong results when the elements of $A^k$ will be large (e.g. the 17-th power of adjacency matrix of the unique SRG with parameters $(36, 10, 4, 2)$ shows on 32-bit computers some negative entries, which is completely wrong). The same problem exists with computing determinants (for the above matrix even on 64-bit computer we see the nonsensical value -351843720888319.81 of determinant).

Python supports arbitrary precision integers, but it has no efficient linear algebra operations with integer matrices and vectors. So we decided to complement the software tools used with NTL [15], a C++ library for manipulating arbitrary length integers, and for vectors, matrices, and polynomials over such integers. We used the *ctypes* Python module (see [16]) to call the functions in NTL library in Python.

We verified all "small groups" (with at most 227 members) from Table 1, using only basic algorithm applied to local complements. But the testing for three "big" groups of SRG-s was very time-consuming, so we considered to make preliminary classification of nonisomorphic graphs with the same parameters, using some cheap graph invariants.

In Czimmermann [4] there is shown that if two graphs $G, H$ have distinct determinants of their corresponding adjacency matrices, then the spectra of adjacency matrices are also distinct and that two graphs with distinct spectra can be distinguished by the basic variant of algorithm, using the sets of walks between vertices.

The algorithm used for testing of three big groups of SRG-s, i.e. the groups with parameters $(35, 16, 6, 8)$, $(36, 15, 6, 6)$ and $(37, 18, 8, 9)$ can be informally described as follows:

1. *generate the sets of local complements* $\mathcal{S}_G = \{G_L(u), u \in V(G)\}$ *for all graphs* $G$ *in the group of nonisomorphic graphs with the same parameters*

2. *compute the determinants of adjacency matrices of graphs in* $\mathcal{S}_G$*, order the sequence of determinants by magnitude; next we partition the group of nonisomorphic SRG-s into the equivalence classes of graphs with the same (ordered) sequence of determinants of local complements; the graphs belonging to a class containing more than one member are indistinguishable by this graph invariant*

3. *now it is sufficient to apply the basic variant of algorithm only for testing isomorphism of pairs of graphs, contained inside the same equivalence class.*

The group of 3854 graphs with parameters $(35, 16, 6, 8)$ has 44 equivalence classes with more than one member, 42 of them have exactly two members and there are two classes with four members. In the group of 32548 graphs with parameters $(36, 15, 6, 6)$ we found 160 equivalence classes (152 with two members, 6 with three members and 2 with four members). For the last group of 6760 graphs with parameters $(37, 18, 8, 9)$ there are 3379 equivalence classes, each with two members. The basic algorithm applied to such small groups of graphs is really computationally inexpensive.

## 4 Conclusions

The presented algorithm for isomorphism testing can efficiently recognize all known small nonisomorphis SRG-s, thus there is a hope it will perform well also on bigger strongly regular graphs. This needs further investigation.

## References

[1] T.F. Hain, G. Goldbogen: *Determination of Graph Isomorphism by the Greatest Characteristic String Invariant*, 41-st Annual ACM Southeast Conference, Savannah, Georgia, March 7-8, 2003

[2] P. Czimmermann: *Návrh algoritmu na hľadanie izomorfizmu grafov,* Zborník 6. vedeckej konferencie doktorandov, Nitra, 2005

[3] P. Czimmermann: *On Certain Algorithm for Graph Isomorphism Problem,* Journal of Information, Control and Management Systems, Vol. 3 (2005), No. 1

[4] P. Czimmermann: *Algebraic approach to Graph Isomorphism Testing,* Journal of Information, Control and Management Systems, Vol. 4 (2006), No. 1 (in preparation)

[5] Ch. Godsil, G. Royle: *Algebraic Graph Theory,* Springer-Verlag, New York, 2001

[6] D. Emms, E. R. Hancock, S. Severini, R. C. Wilson: *A dynamical transformation of strongly regular graphs,* preprint arXiv quant-ph/0505026v1, 2005, found on *http://www.arxiv.org*

[7] SRG database on *http://cs.anu.edu.au/˜bdm/data/graphs.html*

[8] Ted Spence: *Strongly Regular Graphs on at most 64 vertices,* can be obtained from *http://www.maths.gla.ac.uk/˜es*

[9] R. Hipp: *SQLite – SQL database engine,*
homepage at *http://www.sqlite.org*, version 3.2.7, 2005

[10] G. Häring: *pysqlite – Python interface to SQLite,*
homepage at *http://pysqlite.org*, version 2.0.5, 2005

[11] W. Ch. Chen: *Hierarchy of Graph Isomorphism Testing,* Technical Report, Dept. of Computer Science, California Institute of Technology, 1984

[12] G. van Rossum: *Python documentation,*
available on *http://www.python.org/doc*

[13] E. Jones, T. Oliphant, P. Peterson and others: *SciPy: Open Source Scientific Tools for Python,* URL *http://www.scipy.org/*, version 0.4.4, 2005

[14] F. Perez: *IPython – an enhanced Interactive Python shell,*
URL *http://ipython.scipy.org*, version 0.7, 2006

[15] V. Shoup: *NTL – a library for doing number theory,*
URL *http: //www.shoup.net/ntl/index.html*, version 5.4, 2005

[16] T. Heller:*Ctypes Python module,*
URL *http://starship.python.net/crew/theller*, version 0.9.6, 2005